Test Report

This document describes the runs of the test generator and test executor that was a part of our DiemTwins implementation.

Test Generator Config

In this section, we describe the parameters of our test generator.

Parameter	Input type	Description
N_VALIDATORS	Integer	Number of unique validators in testcase
N_PARTITIONS	Integer	Number of network partitions per round in the testcase
ALLOWED_LEADER_TYPE	A string. One of {ALL, NON-FAULTY, FAULTY}	What type of validators can be chosen as a leader in the test cases. Used as an enumeration limit
N_ROUNDS	Integer	Number of rounds to be executed per test case
IS_DETERMINISTIC	Boolean	Whether to enumerate leader partitions over N_ROUNDS deterministically while generating test cases
IS_WITH_REPLACEMENT	Boolean	Whether to repeat the same leader partition scenario in a particular test case
N_TESTCASES	Integer	Enumeration limit for number of test cases
TEST_FILE_BATCH_SIZE	Integer	Number of test cases written per test file
N_TWINS	Integer	Number of twins per test case
INTRA_PARTITION_DROP_TYPES	A list containing a subset of {Proposal, Vote, Timeout}. The maximum length is 2.	For what all message types to generate scenarios where we drop messages intra-partition. Reducing the size of this list reduces the number of enumerations

GENERATE_VALID_PARTITION	Boolean	Whether to always generate partition scenarios where there is a super-majority partition(2f + 1 unique replicas)
SEED	Integer	Seed for all random number generations

The test cases generated are being saved in the folder: <project_path>/testcases with the name format as : “testcases_batch_<batch_num>.

Test executor config

In this section, we describe the configuration used by the test executor. This is the output of the test cases generator by the test generator.

Test generator runs

Note: for every run of the test generator, the corresponding config file and generated test cases are present in the ‘generator_files/generator_run_{run_id}/’ directory, where run_id is 1, 2, 3 etc.

Generator run 1

Configuration format used:

```
generator_config = {
   "n_validators": 4,
   "n_partitions": 2,
   "allowed_leader_type": "FAULTY",
   "n_rounds": 4,
   "is_deterministic": True,
   "is_with_replacement": False,
   "n_testcases": 50,
   "test_file_batch_size": 20,
   "n_twins": 1,
   "intra_partition_drop_types": [],
   "generate_valid_partition": False,
   "seed": 12345
}
```

Test Cases generated without any drop configurations:

ID	Nodes	Twins	Partitions	Round	Leader selection	is_deterministic	Generate valid partitions	Test Cases	Total time
1	4	1	2	4	Faulty	False	False	50	0.001 secs
2	4	1	2	4	Faulty	False	True	50	0.001 secs
3	4	1	2	4	Faulty	False	True	1000	0.020 secs
4	4	1	2	4	Non-faulty	True	False	50	0.069 secs
5	4	1	2	4	Faulty	True	False	32760 (all)	0.820 secs
6	4	1	2	4	Non-faulty	True	False	32760 (all)	1.049 secs
7	4	1	3	4	Faulty	True	False	50000	1.277 secs
8	4	1	3	7	Faulty	True	False	50000	1.836 secs
9	7	2	2	7	Faulty	False	False	50000	2.061 secs

Test Cases generated with drop configurations:

Nodes	Twins	Partitions	Round	Leader selection	is_deterministic	Generate valid partitions	Drop type	Test Cases	Total time
4	1	2	4	Faulty	True	False	vote	50	0.008 secs
4	1	2	4	Non-faulty	True	True	vote	50	1.164 secs
4	1	2	4	Faulty	True	False	Vote, proposal	32760 (all)	1.140 secs

Test executor runs

Executor run 1

Executor configuration: Output of generator run 1
Liveness bound: All validators complete 7 rounds and commit the ‘no-op’ proposed in round 5
Injected bugs: None. Code is the Loyal Byzantine Generals implementation of DiemBFT with the correctness defects mentioned in the Phase 4 document fixed.

Test output and analysis

Execution time	12 seconds
Safety check	47/50 passed
Liveness check	18/50 passed

3 of our test cases failed the safety check despite any injected error. Furthermore, around 15 test cases did not execute because the previous runs of test executor got blocked. On analysing the root cause of the errors, we identified that DiemBFT was accepting multiple votes from the same validator(a scenario that happens when a validator and its twin votes on the same block), which was causing commits to take place without an actual quorum, and leading to safety and liveness failures down the line.

Executor run 2

Executor configuration: Output of generator run 1
Liveness bound: All validators complete 7 rounds and commit the ‘no-op’ proposed in round 5
Injected bugs: None. Code is the one used in executor run 1, with the defect in BlockTree’s process vote messages fixed.

Test output and analysis

Execution time	15 seconds
Safety check	50/50 passed
Liveness check	18/50 passed

On fixing the correctness defect present in the DiemBFT implementation, we no longer got safety failures for the testcase. However, a majority of the test cases were failing the liveness check.

We identified two issues that could cause this. Since we are not assured to have super-majority quorums in all testcases, for many of the runs, the validators will never form a TimeoutCertificate, and hence fail the liveness check by default. Another issue is a race condition in our sync-up mechanism. As we were unable to prevent DiemBFT to asynchronously process other messages while waiting for sync-up, we have a race condition where if we receive a proposal message in the middle of sync up, our validator can become stuck.

Executor run 3

Executor configuration: Output of generator run 2
Liveness bound: All validators complete 7 rounds and commit the ‘no-op’ proposed in round 5
Injected bugs: None. Code is the one used in executor run 1, with the defect in BlockTree’s process vote messages fixed.

Test output and analysis

Execution time	17 seconds
Safety check	48/50 passed
Liveness check	36/50 passed

When we required that every test case had a super-majority quorum in all rounds(test generator run 2), we saw a significant improvement of our validators passing the liveness check. However, 2 out of 50 test cases failed the safety check.

On analysing why the safety check failed, we identified that the race condition in our sync-up mechanism(which we identified in executor run 3) also causes blocks that are synced up to commit in the wrong order in the scenario when a proposal message is processed while sync up is happening.

Executor run 4

Executor configuration: Output of generator run 2
Liveness bound: All validators complete 7 rounds and commit the ‘no-op’ proposed in round 5
Injected bugs: Quorum size changed to 2f

Test output and analysis

Execution time	16 seconds
Safety check	24/50 passed
Liveness check	33/50 passed

As expected, when the quorum size is changed to 2f, we have a significant number of cases where safety checks fails. It is interesting to note that even with 50 test cases over 4 rounds, we detected a large number of safety errors.

Executor run 5

Executor configuration: Output of generator run 2
Liveness bound: All validators complete 7 rounds and commit the ‘no-op’ proposed in round 5
Injected bugs: Voting on multiple proposals in the same round(Twins paper section 7.1)

Test output and analysis

Execution time	16 seconds
Safety check	47/50 passed
Liveness check	19/50 passed

When we allow validators to vote on multiple proposal messages in the same round, we have a lot of testcase failures as expected. It is interesting to see that there are a small number of safety failures, but a large number of liveness failures. This may be due to multiple QCs getting formed in the same round, which leads to subsequent rounds getting stuck, and leading to liveness failures.

Executor run 6

Executor configuration: Output of generator run 3
Liveness bound: All validators complete 7 rounds and commit the ‘no-op’ proposed in round 5
Injected bugs: None

Test output and analysis

Execution time	Timed out
Safety check	322/339 passed
Liveness check	115/399 passed

We executed a test executor for 1000 test cases. However, the test executor crashed after executing 339 scenarios.